# Lecture 1 Additional Notes

**Important Terminology:**

* **IDE:** Integrated development environment
  + Allows us to rapidly develop programs.
  + We use an IDE to make our lives easier during development with integrated access to a SDK.
  + Example: Net beans, eclipse, IntelliJ
* **SDK:** Software development kit
  + Software for developing running programs.
* **JDK:** Java Development Kit
  + The software for developing and running Java programs.
  + Consists of a set of separate programs, each invoked from the command line to develop and test java programs.
* **API:** Application programming interface
  + Also known as a library, which contains predefined classes and interfaces for developing programs.
* **JVM:** Java Virtual Machine
  + Allows us to run bytecode on any platform having the JVM installed.
  + Runs on top of the physical platform/OS
* **Bytecode:** Similar to machine instructions but is architecture neutral and can run on any platform which has the JVM installed.

**The Three Types of Programming Errors:**

1. **Syntax Errors:** 
   1. Detected by compilers
   2. Result in code construction problems
   3. Easy to detect as you will not be able to run the program with a syntax error
   4. Example: Missing a semi-colon
2. **Runtime Errors:**
   1. Occurs at the time the program is running
   2. Cannot be detected by the compiler
   3. Typically rises when an operation is impossible to carry out
   4. Example: Trying to divide by 0
3. **Logic Errors:**
   1. Occurs when a program does not provide a desired output
   2. This is on the behalf of the programmer who made a mistake in the logic of the program
   3. Example: The program was supposed calculate distance in kilometers but the formula for was miles

**The Process of Creating, Compiling, and Running a Java Program:**

1. Create/Modify Source Code
2. Save source code on disk
3. Compile source code into byte code
   1. Accomplished through JDK compiler
   2. Creates a .class file
4. Byte code is then stored on disk
5. Run byte code
   1. Accomplished through an interpreter (Aka the JVM)
   2. Can only run on platforms, which have the JVM installed.
6. View program result

### **Reading Binary Numbers:**

* When we read binary numbers, we start at the least significant bit (LSB) which is the farthest bit to the left then read to the most significant bit (MSB) farthest bit to the right.
* Every bit has a base of 2 and its exponent is corresponding to its position in the sequence.
* Example: 1111 in binary would be 15
* 1(2^3) 1(2^2) 1(2^1) 1(2^0) : 2^3 + 2^2 + 2^1 + 2^0 = 15
* Say for instance we have 1 byte of data = 8 bits.
* 8 bits = 0000 0000
* If the byte is unsigned (Only numbers starting from 0 and on):
  + The minimum number we can store is 0.
  + The maximum number we can store is: 255
  + 1111 1111
  + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255
  + To easily say what the max number in this scenario is we write (2^8 – 1) = 255
* If the byte is signed (Any number negative or positive including 0):
  + The minimum number we can store is - (2^7) = - 128
  + The maximum number we can store is (2^7 – 1) = 127
  + 0111 1111 (The most significant bit represents the positive sign)
  + Thus, 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 127
  + To easily say what the max number in this scenario is we write (2^7 – 1) = 127
  + So what if we have 1111 1111?
  + Well, the most significant bit represents the negative sign.
  + Thus, – (2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0) = - 127
  + So why is the minimum number -128? Why isn’t it -127?
  + Well, - 128 = 1000 0000
  + The most significant bit in this case is special. It represents both the number and the negation sign.
  + According to the last example, this would indicate the number is - 0.
  + Since there is no such thing as - 0 we take what the number would be in the binary position if it were positive (128) and negate it.

### **Base 2 (The Binary System):**

* A binary bit can be 0 or 1
* 1 Byte = 8 bits
* 1 Kilobyte (KB) = 1024 Bytes = 8 Kilobits (Kb)
* 1 Megabyte (MB) = 1024 Kilobytes = 8 Megabits (Mb)
* 1 Gigabyte (GB) = 1024 Megabytes = 8 Gigabits (Gb)
* 1 Terabyte (TB) = 1024 Terabytes = 8 Terabits (Tb)

### **Base 10 (The Decimal System):**

* A binary bit can be 0 or 1
* 1 Byte = 8 bits
* 1 Kilobyte (KB) = 1000 Bytes = 8 Kilobits (Kb)
* 1 Megabyte (MB) = 1000 Kilobytes = 8 Megabits (Mb)
* 1 Gigabyte (GB) = 1000 Megabytes = 8 Gigabits (Gb)
* 1 Terabyte (TB) = 1000 Terabytes = 8 Terabits (Tb)

**Download Speed Conversion Example:**

* Bytes to Bits: Multiply by 8.
* Bits to Bytes: Divide by 8.
* If I have 150 Mb/s download speed from my Internet Service Provider (ISP) then I can download @ 18.75 MB/s.

### **The External HDD (Hard Disk Drive) Example:**

* So why is my 1 TB HDD 931 GB and not 1024 GB?
* HDD manufacturers market and sell HDD’s according to the base 10 system.
* When we do the conversion from Base 10 bytes to Base 2 bytes, we can see how many bytes of data we actually have.
* 1 TB = 1,000,000,000,000 bytes in Base 10.
* 1 MB = 1,048,576 bytes in Base 2.
* 1,000,000,000,000 / 1,048,576 = 953,674.31 MB
* 953,674.316406 / 1024 = 931.32 GB
* Thus, we miss close to 70 GB due to the conversion.
* The following link will provide a bit more insight about this example.
* <http://knowledge.seagate.com/articles/en_US/FAQ/172191en>

**32-bit Operating System (OS) vs 64-bit Operating System (OS):**

* The primary difference between the two is how much RAM can be utilized by the system.
* In a 32-bit OS the highest number which can be represented by an unsigned integer is 2^32 – 1 = 4,294,967,295 (Remember the -1 is because we include 0 into the set of numbers)
* Since 4 GB = 4,294,967,296 bytes we can create addresses in RAM that range from 0 - 4,294,967,295
* In a 64-bit OS the highest number which can be represented by an unsigned integer is 2^64 – 1 = 18,446,744,073,709,551,615. This means we can have at most 17,179,869,184 GB of RAM.
* **Fun Fact**: With this much RAM we could store (all the data that is currently present on the internet) x 13
* The following video will go into more detail about how RAM works.
* <https://www.youtube.com/watch?v=F0Ri2TpRBBg>